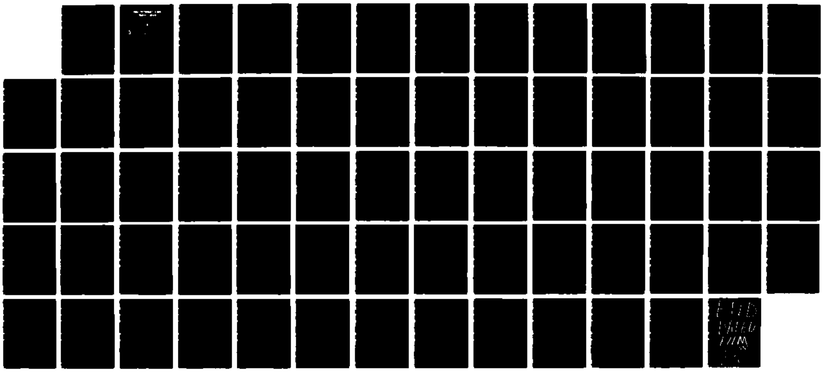
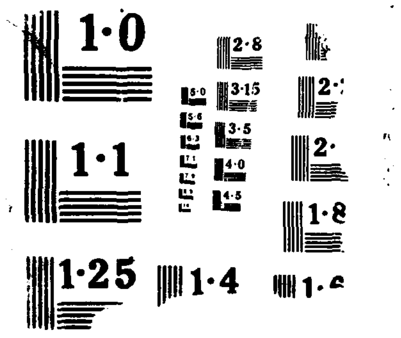
AD-A195 449 SOLUTION OF LARGE-SCALE MULTICOMMODITY NETWORK FLOW 1/1
PROBLEMS VIA A LOGARITHMIC BARRIER FUNCTION
DECOMPOSITION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA
UNCLASSIFIED H LANGE MAR 88 F/G 12/4 NL




1·0
2·8
3·15
3·5
4·0
4·5
1·1
1·25
1·4
1·8

2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SOLUTION OF LARGE-SCALE MULTICOMMODITY NETWORK FLOW PROBLEMS VIA A LOGARITHMIC BARRIER FUNCTION DECOMPOSITION

by

Heinrich Lange

March 1988

Thesis Advisor: R. Kevin Wood

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 55	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
SOLUTION OF LARGE-SCALE MULTICOMMODITY NETWORK FLOW PROBLEMS VIA A LOGARITHMIC BARRIER FUNCTION DECOMPOSITION

12 PERSONAL AUTHOR(S) LANGE, HEINRICH

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1988 March	66

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Multicommodity Network Flow Problem, Large Scale Programming, Logarithmic Barrier Function, Price Directive Decomposition.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A new algorithm is presented using a logarithmic barrier function decomposition for the solution of the large-scale multicommodity network flow problem. Placing the complicating joint capacity constraints of the multicommodity network flow problem into a logarithmic barrier term of the objective function creats a nonlinear mathematical program with linear network flow constraints. Using the technique of restricted simplicial decomposition, we generate a sequence of extreme points by solving independent pure network problems for each commodity in a linear subproblem and optimize a nonlinear master problem over the convex hull of a fixed number of retained extreme points and the previous master problem solution. Computational results on a network with 3,300 nodes and 10,400 arcs are reported for four, ten and 100 commodities. Keywords

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL R. Kevin Wood	22b TELEPHONE (Include Area Code) 408-646-2743 22c OFFICE SYMBOL 55Wd

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Solution of Large-Scale Multicommodity Network Flow Problems via a Logarithmic Barrier Function Decomposition

by

Heinrich Lange
Lieutenant Commander, Federal German Navy
Diplom Kaufmann, German Armed Forces University, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1988

Author: Heinrich Lange
Heinrich Lange

Approved by: [illegible]
R. Kevin Wood, Thesis Advisor

[illegible]
Gerald G. Brown, Second Reader

R R Read (FOR)
Peter Purdue, Chairman
Department of Operations Research

[illegible]
James M. Fremgen
Acting Dean of Information and Policy Sciences

ABSTRACT

A new algorithm is presented using a logarithmic barrier function decomposition for the solution of the large-scale multicommodity network flow problem. Placing the complicating joint capacity constraints of the multicommodity network flow problem into a logarithmic barrier term of the objective function creates a nonlinear mathematical program with linear network flow constraints. Using the technique of restricted simplicial decomposition, we generate a sequence of extreme points by solving independent pure network problems for each commodity in a linear subproblem and optimize a nonlinear master problem over the convex hull of a fixed number of retained extreme points and the previous master problem solution. Computational results on a network with 3,300 nodes and 10,400 arcs are reported for four, ten and 100 commodities.

TABLE OF CONTENTS

I. INTRODUCTION

Multicommodity network flow problems emerge when several distinct commodities flow through a common capacitated network and share one or more arcs that are subject to joint capacity constraints. The objective is usually to find the minimum cost flow given demands and supplies of the commodities. Problems of this type are also referred to as "multicommodity capacitated transshipment problems" and "multicommodity flow problems" (MCFPs).

The problem of optimizing MCFPs arises frequently in logistic systems. As long as only a single commodity is involved, even large-scale problems can now be solved routinely by specialized network codes that exploit the pure network structure of the problem (e.g., GNET by Bradley, Brown, and Graves [Ref. 1]). Such solvers are not directly applicable to the general MCFP, and general linear program solvers are usually inapplicable as a result of the large constraint matrix encountered with typical MCFPs.

Because of the importance of the MCFP, much effort has been devoted to finding efficient, specialized solution techniques. Earlier surveys are given by Kennington [Ref. 2] and Assad [Ref. 3]. New and effective decomposition methods were recently developed by Staniec [Ref.4] and show encouraging results. This paper extends that research by deriving and implementing a decomposition for the MCFP based on a logarithmic barrier function.

A. STATEMENT OF THE PROBLEM

In order to formulate the MCFP as a mathematical program the following notation is used :

G = $\{I,J\}$ is a network with set of nodes I and set of arcs J.

P is the set of commodities (products) flowing on G.

i $\in I$ is a node of G.

j $\in J$ is an arc of G.

p $\in P$ is a commodity flowing through G.

N_p is an $|I| \times |J|$ node-arc incidence matrix for each product ($N_1 = N_2 = \ldots = N_{|P|}$).

N is an $|I|\cdot|P| \times |J|\cdot|P|$ matrix with matrices N_p along the diagonal, 0s elsewhere.

A is a $|J| \times |J|\cdot|P|$ matrix $(I,I,\ldots,I)$.

c $= (c_1,\ldots,c_{|P|})$ is a vector of arc costs, length $|J|\cdot|P|$.

x $= (x_1,\ldots,x_{|P|})$ is a vector of arc flows, length $|J|\cdot|P|$.

b_1 is a vector of joint capacities with length $|J|$.

$\bar{b}_1$ is the vector $(b_1,\ldots,b_1)$ with length $|J|\cdot|P|$.

b_2 is the vector of supplies and demands for each commodity with length $|J|\cdot|P|$.

Then the MCFP may be stated as follows :

$$
\begin{array}{llll}
(P) & \min & cx & \text{(duals)} & (1) \\
& \text{s.t.} & Ax \le b_1 & (u_1) & (2) \\
& & Nx = b_2 & (u_2) & (3) \\
& & 0 \le x \le \bar{b}_1 & (u_3) & (4)
\end{array}
$$

For the constraint sets (3) and (4) the abbreviated notation $x \in F$ will be used.

The stipulation that binds the flow of several commodities to joint capacities is given by (2) and constitutes the set of "complicating constraints". Without those constraints, the problem would reduce to independent, bounded, single-commodity flow problems. The duals u_1 corresponding to (2) are nonpositive. The constraints (4) are redundant, but they ensure that x is bounded when the constraints in (2) are relaxed.

B. SOLUTION METHODOLOGY

The two basic approaches to solving the MCFP (other than a standard primal linear programming method) can be characterized as either decomposition or partitioning techniques. The latter employ a special basis factorization within a simplex algorithm in such a way that portions of each generated basis maintain characteristics of the pure network flow problem. Those method are not investigated here. For further detail see Kennington and Helgason [Ref. 5].

Decomposition methods solve MCFP by using a master problem that coordinates the solution of single network flow subproblems. These methods are attractive, since they may require the internal storage of only one commodity at a time. This approach can further be divided into resource-directive and price-directive algorithms. Both will be stated here for later reference.

The resource-directive method uses a master problem that distributes improving capacity allocations to the individual commodity subproblems. For this purpose MCFP may be written as

$$\begin{array}{llr} (P') & \min_{y} \min_{x} \; cx & (5) \\ \text{s.t.} & Ay = b_1 & (6) \\ & x - y \le 0 & (7) \\ & Nx = b_2 & (8) \\ & 0 \le y \le \bar{b}_1 & (9) \\ & 0 \le x \le \bar{b}_1 . & (10) \end{array}$$

Any vector y satisfying constraints (6) and (9) may be interpreted as a "capacity allocation" which apportions the capacity of an arc across the individual commodities. The inner minimization for a fixed y amounts to a restriction of (P). If its solution is feasible in (P), it yields an upper bound on the optimal solution.

The subproblems for a particular allocation vector $\hat{y}$ are of the form

$$(SP1(\hat{y})) \qquad \bar{V}(\hat{y}) = \min cx \tag{11}$$

$$\text{s.t.} \qquad Nx = b_2 \tag{12}$$

$$0 \leq x \leq \hat{y} \tag{13}$$

which is a set of single-commodity minimum cost flow problems, solved independently for each commodity.

The master problem than becomes

$$(MP1) \qquad \min V(y) \tag{14}$$

$$\text{s.t.} \qquad Ay \leq b_1 \tag{15}$$

$$0 \leq y \leq \bar{b}_1 . \tag{16}$$

The objective function in (14) is piecewise linear and convex. In theory, the master problem can be solved by subgradient optimization or by a cutting plane algorithm.

Penalty and barrier function decomposition are examples of price-directive decomposition. Before describing them, it is worthwhile looking at the Lagrangian relaxation of (P). If the joint capacity constraints are placed into the objective function with multipliers $u_1 \leq 0$, the Lagrangian dual of (P) is

$$(LR) \qquad \max_{u_1 \leq 0} \ \min_{x \geq 0} L(u_1, x) = cx + u_1(b_1 - Ax) \tag{17}$$

$$\text{st.} \qquad x \in F.$$

This problem corresponds to solving $\max_{u_1 \leq 0} LR(u_1)$ where $LR(u_1)$ is defined by

$$(LR(u_1)) \qquad \min_{x \in F} cx - u_1(Ax - b_1)$$

$$= \min_{x \in F} (c - u_1 A)x + u_1 b_1 . \qquad (18)$$

Note that the evaluation of $LR(u_1)$ requires the solution of $|P|$ independent single commodity problems. Furthermore, for any fixed $u_1 \leq 0$, $LR(u_1)$ yields a lower bound on (P) and this bound will be tight if u_1^* is optimal in (P): $LR(u_1^*) = cx^*$.

$LR(u_1)$ is a piecewise linear and concave function that can be optimized, in theory, by subgradient optimization (for example, see Fisher [Ref. 6], Goffin [Ref. 7] or Sandi [Ref. 8]), or by a cutting plane method (see Kelly [Ref. 9]). Optimization of $LR(u_1)$ by a cutting plane algorithm is essentially equivalent to solving the MCFP by Dantzig-Wolfe decomposition (see Staniec, [Ref. 4]).

However, even if (18) is solved optimally, it is possible that $LR(\tilde{u}_1) = cx^*$ for $\tilde{u}_1 \neq u_1^*$. Furthermore, we may not be able to obtain a corresponding primal solution that is feasible to problem (P).

Solving $LR(u_1)$ for any u_1 generally allows some constraints to be violated with penalty $-u_1(Ax - b_1)$. Other penalty functions are possible which will, at least in a limiting sense, yield optimal primal solutions. The objective function in (P) can be transformed into a nonlinear auxiliary function Q(h,x) that typically includes a polynomial penalty term of the form

$$(PP(h)) \quad \min_{x \in F} \quad Q(h,x) = cx + \frac{h}{t} ||(Ax - b_1)_+||_t^t \qquad (19)$$

$$= cx + q(h,x) \qquad (20)$$

where $|| \cdot ||_t$ indicates the t^{th} norm, and $(Ax - b_1)_+$ is the vector $\max(0, Ax - b_1)$, i.e. the vector of violations of the constraint set (2). The value of h constitutes a positive, increasing sequence of penalty parameters. Furthermore, $t > 1$ is required for this method to ensure convergence.

The penalty function has some attractive properties, as given in Luenberger [Ref. 10] : Let $\{ h^k \}$ and $\{ x^k \}$ be sequences of penalty parameters and optimal solutions to $(PP(h^k))$, respectively, with $h^{k+1} > h^k$ and $h^0 > 0$. Then

$$Q(h^k, x^k) \leq Q(h^{k+1}, x^{k+1}), \qquad (21)$$

$$q(h^k, x^k) \geq q(h^{k+1}, x^{k+1}), \text{ and} \qquad (22)$$

$$cx^k \leq cx^{k+1} \; . \qquad (23)$$

The algorithm converges to the optimal solution. Thus,

$$\lim_{h^k \to \infty} Q(h^k, x^k) = cx^* \; , \text{ and } x^k \to x^* \; .$$

Since a feasible solution is usually not obtained until final convergence occurs, the penalty approach is classified as an exterior point algorithm.

In order to obtain intermediate feasible solutions a suitable scaling or projection procedure may be used. If we model the MCFP to include additional bypass arcs which satisfy any undeliverable demand at a high cost from a supersource, we can assume that any allocation of capacity satisfying (6) and (9) is also feasible in F. One possibility to obtain such capacity allocations $\hat{y}$ from a given vector $\hat{x} \in F$ is given by defining $\hat{y} = CA(\hat{x})$ as

$$\hat{y}_{pj} = \begin{cases} \hat{x}_{pj} b_{1j} / (A\hat{x})_j & \text{if } (A\hat{x}-b_1)_j \geq 0 \\ \hat{x}_{pj} + (b_1 - A\hat{x})_j / |P| & \text{if } (A\hat{x}-b_1)_j < 0. \end{cases} \tag{24}$$

Then, a solution of (SP1($\hat{y}$)) as defined in (24) yields a valid upper bound at the current iterate $\hat{x}$. This allocation procedure was successfully applied by Staniec [Ref. 4].

The idea behind the general barrier function approach is just the opposite of the exterior point penalty methods. Starting with a feasible solution which lies within the interior of the constraint region, a modified objective function establishes a barrier against leaving the feasible region. For the MCFP, the auxiliary function with respect to the joint capacity constraints then becomes

$$(BP(h)) \quad \min_{x \in F} \; B(h,x) = cx + b(h,x). \tag{25}$$

An ideal barrier term b(x) would take the value zero for all interior points and infinity at the boundary. A sequential barrier function b(h,x) omits this discontinuity and may take the following forms :

$$b(h,x) = h\, \Sigma_j\, (b_1 - Ax)_j^{-1} \qquad (26)$$

or

$$b(h,x) = -\, h\, \Sigma_j\, \ln(b_1 - Ax)_j, \qquad (27)$$

where $(b_1 - Ax)_j$ denotes the j^{th} component of the vector $(b_1 - Ax)$.

The first expression (26) is called the "inverse barrier function" and the second term (27) the "logarithmic barrier function" (see, for example, Fiacco and McCormick [Ref. 11] or Bazaraa and Shetty [Ref. 12]). Both functions approach the ideal barrier function b(x) as $h \rightarrow 0$. Any barrier algorithm requires the existence of an interior region, i.e. it does not work for equality constraints.

The logarithmic barrier function was first proposed by Frisch 1955 [Ref. 13]. It was then derived together with the inverse barrier function from the Kuhn-Tucker conditions for optimality by Fiacco and McCormick [Ref. 11]. The logarithmic barrier function has obtained recent attention in linear programming due to its fundamental properties. Megiddo [Ref. 14] investigates the properties of a weighted logarithmic barrier function for general linear programs that places the nonnegativity constraints on x into the barrier term while requiring strictly positive values for x. This approach leads to a smooth path through the interior of the constraint region towards the optimal solution and theoretically validates its use in linear programming. Gill, Murray, Saunders, Tomlin and Wright [Ref. 15] established a close relationship to the projective linear programming algorithms initiated by Karmarkar [Ref. 16].

The transformation of the objective function into a penalty or barrier function creates a nonlinear programming problem which requires an efficient solution method to make such transformation profitable. Staniec [Ref.4] successfully applied the method of restricted simplicial decomposition (RSD) in order to solve the penalty function decomposition. The idea was developed by Hearn, Lawphongpanich and Ventura [Ref. 17]. RSD solves any pseudoconvex optimization problem with linear constraints by generating extreme points in linear subproblems while a master problem optimizes the original objective function over a simplex derived from a fixed number of retained extreme points plus the last iterate, i.e. the last solution to the master problem. The implementation of this method for the MCFP will be described later in more detail since it proves useful for the barrier decomposition as well.

C. TEST PROBLEMS

A real-world large-scale MCFP should be used in order to examine the efficiency of the proposed algorithms. Staniec developed an appropriate problem that describes the transshipment of conventional ammunition from production and storage locations to overseas debarkation points and theatre-of-war locations via capacitated road, rail, sea and air transportation links. The product demands are time-phased and the objective is to minimize the weighted deviation from on-time deliveries. The network contains backlogging arcs with graduated penalties and bypass arcs that satisfy undeliverable demand at high cost. For more details see Staniec [Ref. 4 and 18]. The same network is used to test and compare the algorithms for four, ten, and 100 commodity problems. The underlying network contains 3,300 nodes and 10,400 arcs of which about 10% are subject to non-redundant capacity constraints.

II. THE CONCEPT OF BARRIER FUNCTION DECOMPOSITION

The original MCFP (P) constitutes a linear program. It is only the size of the problem that makes a primal simplex algorithm unattractive and computationally expensive. Penalty and barrier functions were initially designed for nonlinear programming problems where they proved useful in converting a constrained nonlinear optimization problem into an unconstrained nonlinear problem. For the special case of the MCFP, penalty and barrier functions provide good decomposition tools.

The barrier function decomposition for the MCFP retains the basic decomposition idea of the penalty function approach. The overlapping joint capacity constraints in (2) are placed into the barrier term of the objective function in order to enable the successive solutions of independent single commodity problems in a sequence of subproblems. The selection of the logarithmic or inverse barrier function is not arbitrary but has an interesting theoretical derivation.

A. THE DERIVATION OF THE BARRIER FUNCTION

Fiacco and McCormick derive the barrier function from the Kuhn-Tucker sufficiency conditions for constrained minima [Ref. 11]. A good and detailed analysis, including implementations and numerical results, is further given by

Wright [Ref. 19]. The derivation shows that the use of a logarithmic barrier function is not arbitrary since it has a very natural origin.

For the purpose of this analysis it is convenient to restate the problem (P) in the following form :

$$(P'') \qquad \min_{x \in F} \; f(x) = cx \qquad (28)$$

$$\text{s.t.} \qquad g(x) = b_1 - Ax \geq 0 \qquad (29)$$

where the objective function (28) has gradient $\nabla f(x) = c$ and each constraint in (29) has gradient $\nabla g_j(x) = - a^j$, the negative of the jth row in A. We associate the dual variables μ_1 with the constraints in (29) and note that in comparison with the duals u_1 of problem (P), μ_1 now takes the opposite sign : $u_1 = - \mu_1$.

Following the derivation of Fiacco and McCormick, we assume that there exist points in the neighborhood of the optimal solution to (P'') such that strict inequality holds for the constraints in (29) i.e., $g(x) > 0$. Furthermore, we allow a perturbation of magnitude h in the Kuhn-Tucker sufficiency conditions for optimality. At some point $[x(h),u_1(h)]$ near the optimum (x^*,u_1^*) the following conditions have to hold for h small and for all $j \in J$:

$$(b_1 - Ax)_j > 0 \qquad \text{(primal feasibility)} \quad (30)$$

$$\mu_{1j} \, (b_1 - Ax)_j = h > 0 \qquad (31)$$

(perturbed complementary slackness)

$$\mu_{1j} \geq 0 \qquad (32)$$

(dual feasiblity)

$$c - \Sigma_j \mu_{1j} (-a^j) = 0. \qquad (33)$$

Rewriting (31) as $\mu_{1j} = h / (b_1 - Ax)_j$ and substituting in (33) yields :

$$c - h \Sigma_j [(b_1 - Ax)_j]^{-1} (-a^j) = 0. \qquad (34)$$

Using the notation $g_j = (b_1 - Ax)_j \geq 0$ for all $j \in J$, equation (34) is of the general form

$$\nabla B(h,x) = \nabla f(x(h)) - h \Sigma_j \frac{\nabla g_j[x(h)]}{g_j[x(h)]} = 0 \qquad (35)$$

and simply means that the gradient of the objective function

$$B(h,x) = f(x) - h \Sigma_j \ln(b_1 - Ax)_j \qquad (36)$$

vanishes at x(h). This is the logarithmic barrier function! Note that no constraint qualifications are necessary since all constraints in the MCFP are linear.

Fiacco and McCormick *show* that the second order sufficiency conditions are also satisfied for B(h,x) at $[x(h), u_1(h)]$ near the optimum x^* and prove the existence of x(h) satisfying these conditions. It is also shown that the Hessian matrix is positive definite for small h.

The same authors obtain the inverse barrier function from a modification in the derivation above that enforces nonnegativity in (32) by introducing a variable τ such that $\tau^2 = \mu_{1j}$. The logarithmic barrier function seems to be the more fundamental approach.

B. PROPERTIES OF THE LOGARITHMIC BARRIER FUNCTION

The basic properties of the barrier function are again given by Fiacco and McCormick [Ref. 11] as well as by Wright [Ref. 19] and can be stated as follows for the MCFP :
For a decreasing sequence $\{h^k\}$ and associated minima $\{x^k\}$ the following conditions hold :

$$B(h^k, x^k) \leq B(h^{k-1}, x^{k-1}) \tag{37}$$

for sufficiently small h^k and bounded $(b_1 - Ax)_j$, and

$$cx^k \leq cx^{k-1}, \tag{38}$$

$$\Sigma_j \ln(b_1 - Ax)_j^k \leq \Sigma_j \ln(b_1 - Ax)_j^{k-1}, \tag{39}$$

$$\lim_{h^k \to 0} h^k \Sigma_j \ln(b_1 - Ax)_j^k \to 0, \text{ and} \tag{40}$$

$$\lim_{h^k \to 0} B(h^k, x^k) \to cx^*. \tag{41}$$

The following small example will illustrate these properties. The problem consists of a nonlinear objective function and a single constraint : Min $2x^2$ s.t. $x \geq 1$ and has the optimal solution $x^* = 1$. The barrier function

$B(h,x) = 2x^2 - h \ln(x-1)$ has a closed form solution

$x(h) = 0.5 + (0.25 + 0.25\,h)^{0.5}$.

The solutions are shown in Figure 1 for linearly decreasing values of h from 20 to 0.1. The change in x as a function of h and the corresponding logarithmic term are depicted in Figure 2.

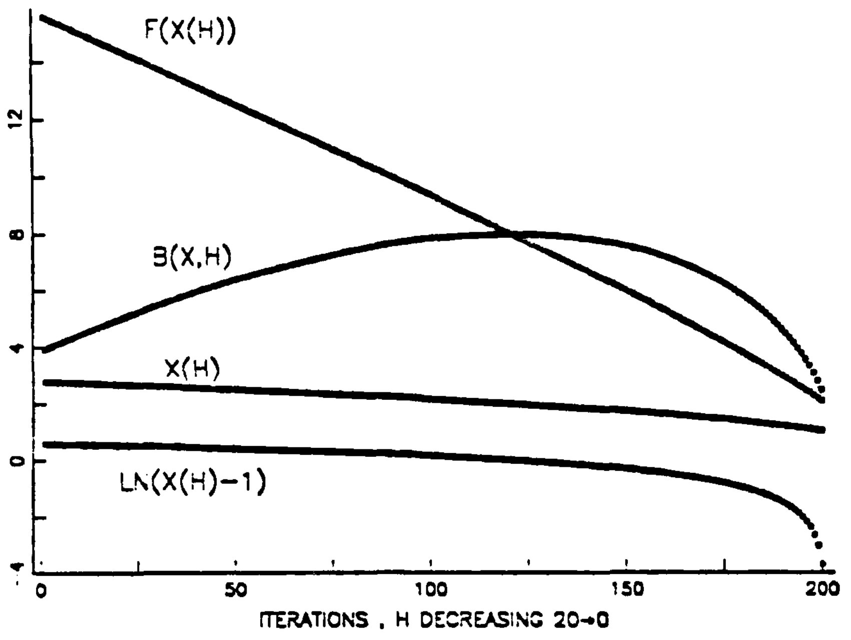


Figure 1 : Trajectory of a Barrier Function Optimization

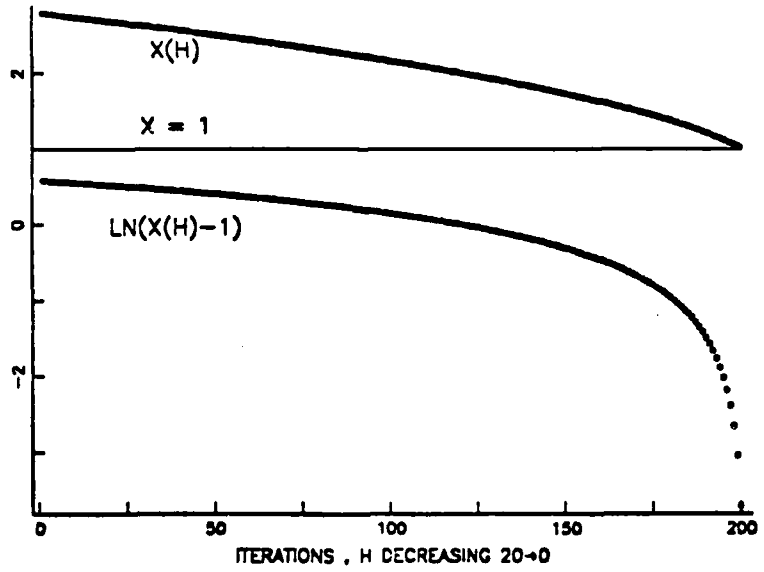


Figure 2 : x(h) and b(h,x) for h decreasing

It is interesting to observe that the path of x(h) towards its boundary $x = 1$ is smooth and can be well approximated by a straight line until $B(h,x)$ reaches its maximum. From that point on the rate of change in $x(h)$ increases slightly. Similar results are obtained for different objective functions. Returning to the multiple constraint barrier function for the MCFP, the maximum of $B(h,x)$ is obtained for a value h' which satisfies the stationarity condition $\Sigma_j \ln(b_1 - Ax(h'))_j = 0$. This can only occur if some components of $(b_1 - Ax(h))$ are less than or equal to one, i.e., some constraints are almost tight. Thus, we will not be able to observe the property in (37) until the final stage of the algorithm when relatively small values of h are attained.

The properties (38) and (41) are the most important ones for practical purposes. Starting with a feasible (interior) point, the algorithm produces a nonincreasing sequence of objective values which converges to the optimal value of (P) and provides intermediate feasible solutions along its path. This path of x as a function of h describes a trajectory that has already been studied by Fiacco and McCormick [Ref. 11] as well as by Wright [Ref. 19]. The existence of this trajectory could be utilized in an extrapolation technique predicting the next iterate or even x^*. Its implementation for the large-scale MCFP is not investigated here. However, we will be able to get a good feasible solution after only a few iterations without using an extrapolation technique.

One final property is the robustness of the barrier function method: Mifflin [Ref. 20] showed that it is sufficient to solve B(h,x) at each iteration only approximately, within a predetermined tolerance, while still achieving convergence (at a lower rate).

C. COMPUTATIONAL CONSIDERATIONS

Any barrier function technique requires an interior starting point. It may not be easy to find a starting point and the performance of an algorithm is influenced by the quality of this initial solution. We utilize the capacity allocation mechanism $\hat{y} = CA(\hat{x})$ for the generation of an initial starting point.

The transformation of (P) into a nonlinear programming problem requires an effective NLP solution methodology. If a line search is part of this method, any discrete-step line search procedure along an improving direction may cause the evaluation of B(h,x) at one or more infeasible points. This requires extra precautions during the implementation, resulting in additional computation time.

Ryan [Ref. 21] points out that for small values of h the auxiliary function B(h,x) becomes very "steep valleyed" and the gradient $\nabla B(h,x)$ can take large values in a small neighborhood of x(h). Therefore, a termination criterion based on the magnitude of the gradient alone may be critical as h becomes small; termination criteria based on the difference in successive objective values may lead to premature termination. We will consider both criteria and take the risk of not solving the problem optimally at each step.

Another well-known difficulty of barrier function approaches is the ill-conditioned Hessian matrix of B(h,x) for small h (see,for example Bazaraa and Shetty [Ref. 12], Wright [Ref. 19] or Ryan [Ref. 21]). This problem emerges only in the final stage as $h \rightarrow 0$. For large-scale programming purposes the achievement of a solutiom which defers from the optimum only by some small value ϵ (ϵ-optimality) is normally sufficient. Such a solution may be obtained before the ill-conditioning becomes bothersome.

Another issue that significantly influences the performance of the algorithm is the choice of the initial barrier parameter h and its rate of decrease. Lootsma [Ref. 22] showed that the absolute difference $|| x(h) - x^* ||$ is of the order $O(h)$ for the logarithmic barrier function. Ryan [Ref. 21] uses this linear relationship to propose a generating relation of the form $h^k = 10^{1-k} h^0$, where $k = 1,2,3,...,$ and h^0 is positive.

The suitable choice of the initial value h^0 is a more critical issue, since theoretically h^0 can take any value greater than zero and up to infinity. Intuitively however, h^0 should depend on the cost and constraint structure of the problem. One possible choice would be to interpret the parameter h as a scaling factor between the cost vector and the set of constraints placed into the objective function. This leads to the suggestion that we achieve initial balance at the starting point x^0 such that $cx^0 = h^0 \Sigma_j \ln(b_1 - Ax^0)_j$.

Instead of analytically deriving h^0, simply taking an multiple σ (σ greater than one) of the maximum cost value in connection with a constant rate of decrease has performed well in test problems :

$h^0 = \sigma * c_{max}$, $h^{k+1} = a * h^k$ for $k > 1$ and $0 < a < 1$.

In general, we recommend choosing h^0 too high rather than too low as the algorithm will more quickly adjust to high values rather than low values.

III. THE LOGARITHMIC BARRIER FUNCTION DECOMPOSITION USING RESTRICTED SIMPLICIAL DECOMPOSITION (RSD)

The basic idea of the barrier function decomposition for the MCFP is to place the coupling constraints (2) into the logarithmic term of the barrier function as derived in Chapter 2. The resulting formulation (BP(h)) constitutes a nonlinear programming problem with linear network flow constraints (3) and (4). Using the restricted simplicial decomposition technique (RSD), we will decompose the problem into a nonlinear master problem and a set of subproblems, which require only the solution of |P| independent pure network flow problems. The master problem has a reduced search space described by a fixed number of retained extreme points, which are generated in the subproblems.

Lower and upper bounds on the optimal solution to (P) can be easily established. The analogy with the penalty decomposition is interesting and worth pursuing. The solution method RSD used in the penalty function decomposition also proves to be effective for (BP(h)) and is described first.

A. RESTRICTED SIMPLICIAL DECOMPOSITION (RSD)

The basic difference between a linear programming problem and a nonlinear programming problem with linear constraints is the fact that the optimal solution will normally not be an extreme point of the constraint region.

The familiar Frank-Wolfe algorithm [Ref. 23] takes advantage of the specialized constraints by generating an extreme point solution of the original problem in a linear subproblem whose objective function is the linearization of the original objective function at the current iterate (given an initial solution). A master problem provides a new iterate via a simple line search between the previous iterate and the new extreme point. The main disadvantage of this decomposition algorithm is its susceptibility to slow convergence, especially when the line search direction becomes orthogonal to the gradient of the objective function (e.g., see Wolfe [Ref. 24]).

The method of simplicial decomposition is due to Geoffrion [Ref. 25], von Hohenbalken [Ref. 26] and Holloway [Ref. 27]. A nonlinear master problem replaces the line search of the Frank-Wolfe method by extending the optimization to the convex hull of all extreme points generated in the linear subproblems. This solution method relies on an effective solution of the nonlinear master problem. Although the master problems have only simple convexity constraints, the increasing number of extreme points makes the implementation of this technique unattractive for the solution of large-scale programming problems.

The disadvantage of simplicial decomposition led to the idea of restricted simplicial decomposition (RSD) as developed by Hearn, Lawphongpanich and Ventura [Ref. 17]. RSD limits the size of the master problem by fixing the maximum number r of retained extreme points. The master problem optimizes over the simplex of these extreme points and the iterate obtained from the previous master solution. Any newly generated extreme point replaces the old extreme point with minimum weight in the expression of the current iterate as a convex combination of the retained extreme points and the prior iterate. After solving the new master problem, all extreme points with zero weight can be discarded.

If r is set to its minimum value $r = 1$, RSD specializes to the algorithm of Frank-Wolfe. For the maximum value of r (the finite number of extreme points) the method represents simplicial decomposition. The solution to the master problem becomes harder as r is increased, but is rewarded by significant improvements in the convergence rate to the optimal solution (see Hearn, et al., [Ref. 28]).

The decomposition of (P'') is formed in the following manner: Let X^k denote a matrix in the k^{th} iteration of the master problem whose columns are a set of r extreme points from F, let $\hat{x}^{k-1}$ be the solution of the previous master problem, and let $\hat{X}^k = X^k \cup \{\hat{x}^{k-1}\}$. The master problem in terms of the weights w at iteration k becomes,

for a fixed h,

$$(\text{MP2}^k) \quad \min B(h,\hat{X}^k w) = c\hat{X}^k w - h\,\Sigma_j \ln(b_1 - A\hat{X}^k w)_j \tag{42}$$

$$\text{s.t.} \quad \mathbf{1}\, w = 1 \tag{43}$$

$$w \geq 0, \tag{44}$$

which has solution $\hat{w}^k$ in terms of w and solution $\hat{x}^k = \hat{X}^k\hat{w}^k$ in terms of x.

The subsequent subproblem optimizes the linear approximation of B(h,x) at $\hat{x}^k$ over F, which is equivalent to :

$$(\text{SP2}^k) \quad \min_{x \in F} \nabla B(h,\hat{x}^k)x\ . \tag{45}$$

It is convenient to introduce the notation $(b_1 - Ax)^t$ to represent the vector $(b_1 - Ax)$ whose components are taken to the t^th power. Then, (SP2^k) can be written as

$$(\text{SP2}^k) \quad \min_{x \in F} (c - h[(b_1 - A\hat{x}^k)^{-1}]^T A)x\ . \tag{46}$$

Using the relationship in (31), we observe that we obtain an estimate $\hat{\mu}_1$ of the optimal dual variables as $\hat{\mu}_1 = h(b_1 - A\hat{x}^k)^{-1}$ at each iteration. Substituting into (46) yields a subproblem as

$$(\text{SP2}^k) \quad \min_{x \in F} (c + \hat{\mu}_1 A)x. \tag{47}$$

This subproblem resembles a standard Dantzig-Wolfe decomposition subproblem as it would be used in a dynamic column generation approach to problem (P) with dual estimates $\hat{u}_1 = -\hat{\mu}_1$. This issue is not discussed here; see Staniec [Ref. 4].

Due to the block structure of the constraint matrix N, ($SP2^k$) permits independent solutions for each commodity. However, each solution to ($SP2^k$) is not necessarily a feasible flow in (P) and of course, is not an interior point. As a starting point, the barrier function approach requires an initial interior point, which may be obtained as follows. First we solve a subproblem with $h = 0$, and yielding solution x^0. Then, using the capacity allocation mechanism $\hat{y} = CA(x^0)$, we further reduce the capacity by setting $\hat{y}' = b * \hat{y}$ for $0 < b < 1$. Finally, we solve subproblem ($SP1(\hat{y}')$) to get an initial interior point solution to (P).

Starting with this solution in the master problem we simply limit our search to that part of the convex hull of extreme points plus the current iterate which provides an interior point solution as next iterate. This will be discussed in connection with the solution of the nonlinear master problem.

B. SOLUTION OF THE MASTER PROBLEM

The reduced gradient method is a well-known approach to solving a nonlinear program with linear constraints and is used here for solving the RSD master problem. The method was first proposed by Wolfe [Ref. 28] and modified by McCormick [Ref. 29]. The basic idea is the partition of the variables x into m basic variables x_B and n nonbasic variables x_N as done in the simplex algorithm of linear programming. This induces a partition of the constraint matrix A into parts B and C, where B is assumed nonsingular. The NLP then takes the form

$$\min f(x) = f(x_B, x_N) \tag{48}$$

$$\text{s.t.} \quad Ax = Bx_B + Cx_N = b \tag{49}$$

$$x_B, x_N \geq 0. \tag{50}$$

The variables x_N are regarded as independent variables whereas x_B are dependent variables completely determined by x_N and equations (49). Consequently, the objective function can be considered to be a function of x_N only and the constraints reduce to the nonnegativity constraints on the independent variables and the limitation in their change that provides nonnegative basic variables. This fact allows the application of a modified steepest descent method accounting only for the nonnegativity constraints. The reduced gradient r^k at iteration k is of dimension n-m and computed as :

$$r^k = ({r_B}^k, {r_N}^k) = \nabla_{x_N} f(x_B^k, x_N^k) - \nabla_{x_B} f(x_B^k, x_N^k)\ B^{-1}C. \tag{51}$$

To find an improving direction d such that $\nabla f(x^k)d < 0$, we select at each iteration k

$$d_{Nj} = \begin{cases} -r_j & \text{if } r_j \leq 0 \\ x_j r_j & \text{if } r_j > 0 \end{cases} \qquad \text{for } x_j \text{ nonbasic} \tag{52}$$

and

$$d_{Bi} = -(B^{-1}Cd_N)_i \qquad \text{for } x_i \text{ basic.} \tag{53}$$

A new direction needs to be computed as soon as a nonbasic variable attains its zero level. If a basic variable becomes zero, the partition must be modified. Also, this method requires a nondegenerate solution at each iteration. A more detailed description can be found in Luenberger [Ref. 10] or Bazaraa and Shetty [Ref. 12].

The use of the reduced gradient method for the solution of the master problem ($MP2^k$) results in some nice simplifications due to the presence of only a single convexity constraint. There is only one basic variable w_i to be selected and the reduced gradient becomes

$$r_j = cx_j - cx_i + h[(b_1 - AXw)^{-1}]^T (Ax_j - Ax_i) \tag{54}$$

for the nonbasic variables w_j. Computing the direction component for the basic variable w_i reduces to

$$d_i = -B^{-1}Cd_j = -\Sigma_j d_j \tag{55}$$

and for the nonbasic variables w_j

$$d_j = \begin{cases} -r_j & \text{if } r_j \le 0 \\ -w_j r_j & \text{if } r_j > 0. \end{cases} \tag{56}$$

The line search in the direction d is limited by the nonnegativity constraints. Thus a maximum steplength α' is computed as

$$\alpha' = \min \{-w_k/d_k \mid d_k \leq 0 \text{ , for all variables } w_k\}. \quad (57)$$

Any move in the direction d of size α must also satisfy

$$b_l - [AX(w + \alpha d)] > 0 \quad \text{for all} \quad 0 \leq \alpha \leq \alpha' \quad (58)$$

or equivalently

$$b_l - AXw > \alpha AXd \quad \text{for all } 0 \leq \alpha \leq \alpha'. \quad (59)$$

Since b_l - AXw > 0 , this holds for all arcs l with $(AXd)_l \leq 0$. If $(AXd)_l > 0$ for some arcs l, we perform a ratio test to select

$$\alpha'' < \min \{ (b_l - AXw)_l \,/\, (AXd)_l \mid (AXd)_l > 0) \quad (60)$$

and set

$$\alpha_{max} = \min \{\alpha', \alpha''\}. \quad (61)$$

The master problem solution is summarized as follows:

Step 0 : Set w such that **1**w =1, $w_k \geq 0$.

Select a basic variable: use the largest w_k.

Step 1 : Compute the direction d as determined in equations (54) through (56). If d = 0, stop.

Step 2 : Solve min $B[h,X(w+\alpha d)]$ s.t. $0 \leq \alpha \leq \alpha_{max}$ in a line search, yielding α_m. Let $w \leftarrow w + \alpha_m d$ and go to Step 1.

The convergence of this method may not be satisfactory since it represents a greedy descent method in the reduced space of the nonbasic variables. Reklaitis et al. [Ref. 30] suggest a convergence acceleration technique by using a more effective unconstrained search method as long as the basic variables do not change. This is especially relevant in our case where the prior iterate frequently takes the largest weight and remains the basic variable. Therefore we apply a conjugate gradient method, that has proven its effectiveness in unconstrained optimization and is easily implemented. The modified direction for the first iteration and any iteration following a basis change becomes :

$$d^k_{Nj} = \begin{cases} - r^k_j & \text{if } w_j > 0 \text{ or } r^k_j < 0 \\ 0 & \text{otherwise,} \end{cases} \tag{62}$$

and for all other iterations :

$$d^k_N = - r^k + \frac{\| r^k \|^2}{\| r^{k-1} \|^2} d_N{}^{k-1} . \tag{63}$$

The direction for the basis variable is the same as in the standard reduced gradient method. The use of the conjugate gradient provides significant improvements. In a typical master problem with only four extreme points, the reduced gradient method used 500 iterations to obtain a solution that was still 13% worse than a solution obtained with the conjugate gradient method in 28 iterations.

Each line search requires frequent evaluations of the objective function and consumes a fair amount of computation time. We can improve this by modifying the objective function to include only those arcs in the barrier term that are potentially able to violate the joint capacity constraints at the current iteration. These arcs are easily identified and recorded in a set J_V containing all arcs that ever had a violation in any of the extreme points generated in the subproblems. The arcs $j \notin J_V$ cannot violate the joint capacity constraints in a solution to the master problem that is a convex combination of the extreme points and the feasible prior iterate. Thus we establish the barrier only on a reduced subset of the joint capacity constraints. This set is updated at each solution of a subproblem in case a new arc experiences a capacity violation. This procedure amounts to a modified barrier function. However, in a finite number of iterations, the number of arcs in J_V will take a fixed value $|J_V| \leq |J|$ and no more arcs will be added to J_V. From this point on we are back to a pure barrier function as derived in Chapter 2 and convergence of the algorithm is preserved.

C. LOWER AND UPPER BOUNDS

Since we will rarely be able to find the optimal solution to (P) within a reasonable number of iterations, we need to establish bounds on the optimal solution.

Lower bounds can be derived from the Lagrangian dual problem

$(LR(u_1))$ $\quad \min_{x \in F} (c - u_1 A)x + u_1 b_1$, which provides a lower bound on (P) for any fixed $u_1 \leq 0$.

Using the dual estimates $\hat{u}_1 = -\hat{\mu}_1$, v.i.z. $\hat{u}_1 = -h\ (b_1 - A\hat{x})^{-1}$ for each solution $\hat{x}$ provided by the master problem, we obtain

$$(LR(\hat{\mu}_1)) \qquad \min_{x \in F} \ (c + \hat{\mu}_1 A)x - \hat{\mu}_1 b_1 . \qquad (64)$$

Recalling the subproblem (SP2^k): $\min_{x \in F} (c + \hat{\mu}_1 A)x$, we find by comparison that both objective functions differ only by the constant term $\hat{\mu}_1 b_1$ and yield the same optimal solution x^k. Thus, we obtain a valid lower bound $\underline{V}(\hat{\mu}_1{}^k)$ by subtracting the constant term $\hat{\mu}_1 b_1$:

$$\underline{V}(\hat{u}_1{}^k) = (c + \hat{u}_1{}^k A)x^k - \hat{\mu}_1{}^k b_1 . \qquad (65)$$

Furthermore, since in the limit $x^k \to x^*$ and $\hat{\mu}^k \to \mu_1{}^*$, it must be that $\underline{V}(\hat{\mu}_1{}^k) \to (c + \mu_1{}^* A)x - \mu_1{}^* b_1 = cx^*$.

Upper bounds on the optimal solution are generated in each master problem, since we restrict the solution to an interior, feasible point. Thus, if $\hat{x}^k = \hat{X}^k \hat{w}^k$ solves (MP2^k), the upper bound

$$\overline{V}(\hat{x}^k) = c\hat{x}^k \qquad (66)$$

is readily available at each master problem solution. Due to the convergence property (41) of the barrier function, it must be that $\overline{V}(\hat{x}^k) \to cx^*$ as $h \to 0$.

However, we are usually able to obtain a better upper bound by utilizing the capacity allocation mechanism again. Let $\hat{y} = CA(\hat{x}^k)$ at some iteration k and let $\hat{x}^k$ be an interior point. Then, for all arcs $j \in J_V$, $(Ax - b_1)_j < 0$ and $\hat{y}_{pj} \geq \hat{x}_{pj}{}^k$ for all p, j from equations (24). Then the following relationship must hold:

$$\overline{V}(\hat{x}^k) = c\hat{x}^k \geq \min_{\substack{x \in F \\ x \leq \hat{x}^k}} cx \geq \min_{\substack{x \in F \\ x \leq \hat{y}}} cx = \overline{V}(\hat{y}),$$

since $\hat{y} \geq \hat{x}^k$.

Thus, solving SP1($\hat{y}$) yields a feasible solution with value $\overline{V}(\hat{y}) \leq \overline{V}(\hat{x}^k)$.

D. THE ALGORITHM RSD(B)

The algorithm RSD(B) using a barrier function decomposition can now be presented. An initial lower bound is obtained by solving the problem without the joint capacity constraints (2). If this solution is feasible in (P), it is optimal. Otherwise we obtain a feasible solution via the capacity allocation mechanism, which gives an initial upper bound. The algorithm generates a sequence of extreme points in the subproblem and an interior point solution in the master problem until ϵ-optimality is achieved. As a heuristic, we invoke the capacity allocation mechanism at every rth iteration of the master problem to improve the current upper bound and decrement h at this time, even if the master problem is not solved optimally for the current h.

Notation :

X^k	matrix of retained extreme points at iteration k
$\hat{x}^k$	current master problem solution
x_M	previous master problem solution
$\hat{X}^k$	matrix X^k augmented with x_M
x^k	optimal solution to subproblem ($SP2^k$)
H(X)	convex hull of X
CA(x)	a capacity allocation based on x
J_V	set of joint capacity arcs which are violated in at least one subproblem solution x^k.
h^l	barrier parameter used in lth parameter update
r	maximal number of retained extreme points
ϵ	stopping criteria for near-optimality.

Algorithm RSD(B)

Input : The network G = {I,J}, joint capacity vector b_1, cost vector c and supply/demand vector b_2.

Output : Best obtained solution $\hat{x}^k$ and final bounds $\overline{V}$, $\underline{V}$.

Step 0 : (Initialization)

```
    Select $h^0 > 0,\ \epsilon > 0,\ r \geq 1,\ 0 < b < 1$.
    Set $k = 0,\ l = 0$,
    $X^0 = \emptyset\ ,\ J_V = \emptyset\ ,\ \hat{\mu}_{1j}^{0} = 0$ for all $j \in J$.
    Solve (LR(0)) : $\hat{x}^0 = \underset{x}{\mathrm{argmin}}\ \{cx \mid x \in F\}$.
    Set $\underline{V} = c\hat{x}^0$.  Set $J_V = \{j \mid (b_1 - A\hat{x}^0)_j < 0\}$.
```

```
        If $J_V = \emptyset$, stop with $\hat{x}^0$ optimal.
        Else, set $\hat{y} = CA(\hat{x}^0)$, $\hat{y}' = b * \hat{y}$ and
        solve SP1($\hat{y}'$) yielding $x_y^0$ .
        Set $\overline{V} = cx_y^0$.  If $(\overline{V} - \underline{V})/\overline{V} < \epsilon$, exit with $x_y^0$, $\overline{V}$, $\underline{V}$.
        Else, set $X^1 = \emptyset$ , $x_M = \hat{x}^1 = x_y^0$ , $k = 1$.

Step 1 : (Solve SP2$^k$)
        Solve $x^k = \operatorname{argmin}_x \{(c + \hat{\mu}_1^k A)x \mid x \in F\}$,
        where  $\hat{\mu}_{1j}^k = h^l(b_1 - A\hat{x}^k)_j^{-1}$  for all $j \in J_V$ and
               $\hat{\mu}_{1j}^k = 0$ for all $j \notin J_V$.
        Set $J_V = J_V \cup \{j \mid (b_1 - Ax^k) < 0\}$.
        Set $\underline{V}(\hat{\mu}_1^k) = (c + \hat{\mu}_1^k)x^k - \hat{\mu}_1^k b_1$.
        If $\underline{V}(\hat{\mu}_1^k) < \underline{V}$, set $\underline{V} = \underline{V}(\hat{\mu}_1^k)$.
        If $(\overline{V} - \underline{V}) / \overline{V} < \epsilon$, exit with $\hat{x}^k$, $\overline{V}$, $\underline{V}$.
        If $(c + \hat{\mu}_1^k A)(x^k - \hat{x}^k) \geq 0$, $\hat{x}^k$ solves $B(h^l, x)$.
                Set $h^{l+1} = a * h^l$  $(0 < a \leq 1)$.
                Set $l = l + 1$ and go to Step 2.
        Else
            (i)  if $|X^k| < r$ , $X^{k+1} = X^k \cup \{x^k\}$.
            (ii) if $|X^k| = r$ , drop the column of $X^k$ which had
                 the smallest weight $\hat{w}_i^k$ in the convex com-
                 bination forming $\hat{x}^k$ and replace it with $x^k$.
                 Go to Step 2.
```

```
Step 2 : (Solve the master problem $MP2^k$)
        Set $\hat{X}^{k+1} = X^{k+1} \cup \{x_M\}$.
        Find $\hat{x}^{k+1} = \underset{x}{\operatorname{argmin}} \{ B(h^l,x) \mid x \in H(\hat{X}^{k+1}) \}$
                  $= \Sigma_i \hat{w}_i^{k+1} x_i$ where $1 \le i \le |\hat{X}^{k+1}|$ and
                        $x_i$ is the $i^{th}$ column in $\hat{X}^{k+1}$.
        Set $x_M = \hat{x}^{k+1}$.
        If $c\hat{x}^{k+1} < \overline{V}$ , set $\overline{V} = c\hat{x}^{k+1}$.
        If $(\overline{V} - \underline{V}) / \overline{V} < \epsilon$, exit with $\hat{x}^k$, $\overline{V}$, $\underline{V}$.
        If k+1 is an integer multiple of r, do a capacity
            allocation : Set $\hat{y} = CA(x^{k+1})$ and solve $SP1(\hat{y})$,
            yielding $\overline{V}(\hat{y}) = cx_y{}^{k+1}$.
            If $\overline{V}(\hat{y}) < \overline{V}$, set $\overline{V} = \overline{V}(\hat{y})$.

            If $(\overline{V} - \underline{V}) / \overline{V} < \epsilon$, set $\hat{x}^k = x_y{}^{k+1}$ ,
                                      exit with $\hat{x}^k$, $\overline{V}$, $\underline{V}$.
            Set $h^{l+1} = a * h^l$ $(0 < a \le 1)$ and $l = l + 1$.
        Set k = k + 1 and go to Step 1.
```

C. IMPLEMENTATION OF RSD(B)

The algorithm RSD(B) has been coded in FORTRAN which is still an extremely efficient language for mathematical programming purposes (see MacLennan [Ref. 31]). A sophisticated data structure is used for the storage of sparse matrices and vectors. Allowing direct communication with the X-system solver of Brown and Graves [Ref. 32]. Integer arithmetic is performed to the extent possible, especially for the subproblems which use a GNET solver [Ref. 1] to produce rapid solutions.

A design goal was set to provide a decomposition procedure which solves only a single commodity problem at a time, and thus operates easily within a modest memory region (say, two megabyte virtual storage capacity). Other information such as current incumbent, previous iterate, prior extreme points, etc., have to be kept on external storage devices. This approach leads to considerable input/output operations at the expense of computation time, but the maximum problem size in terms of number of commodities remains independent of the available virtual storage. Also, the resulting algorithm is highly parallel by commodities.

The implementation of the master problem contains several parameters such as the number of retained extreme points, stopping criteria for optimality at each iteration, the final interval of uncertainty in the line search and an upper limit on the maximum number of line searches conducted. Furthermore, the weights of the extreme points and the objective function evaluation are subject to roundoff errors. For a sensitive objective function, special care is necessary to insure convergence. Fortunately, we will confirm that the barrier function decomposition is a robust procedure that does not require an optimal solution to the master problem at each iteration. Good results are obtained over a relative broad range of parameters.

The comparison of the barrier algorithm RSD(B) with the penalty algorithm RSD(P) of Staniec [Ref. 4] reveals that they are very similar in their sequential structure. This similarity permits embedding both algorithms in the same computer program and creates the potential for devising a hybrid algorithm which takes advantage of each. The relationship between RSD(B) and RSD(P) is easily established. The dual estimates obtained from the penalty approach take, for some vector x and joint capacity constraint j, the form

$$\hat{\mu}_{1j}^{P} = h\ [(Ax - b_1)_+^{t-1}]_j^T \quad , \ t > 1 \qquad (67)$$

versus

$$\hat{\mu}_{1j}^{B} = h[(b_1 - Ax)^{-1}]_j^T \ , \quad (b_1 - Ax)_j > 0 \ , j \in J_V \qquad (68)$$

for the barrier approach. However, the gradient of the penalty function at some vector x takes the same form as for the barrier function, namely

$$\nabla Q(h,x) = c + \hat{\mu}_1^P A \qquad (69)$$

versus

$$\nabla B(h,x) = c + \hat{\mu}_1^B A. \qquad (70).$$

Thus, besides the fact that the barrier function decomposition requires an initial interior point, both methods use the same subproblems and master problem with different input for the dual estimates and different function evaluation routines in the line search.

IV. COMPUTATIONAL RESULTS

In order to assess the capabilities of the algorithm RSD(B), we solve different versions of the test problem described in Chapter I. This problem suite has been extensively studied by Staniec [Ref. 4], using different algorithms. The optimal solution for four and ten commodity problems is available for comparisons, obtained by solving the problem (P) using the X-system [Ref. 32]. The four commodity problem (4H) has approximately 13,200 constraints, 41,600 variables and optimal solution value 130,739,585. The ten commodity problem (10H) has about 33,000 constraints, 104,000 variables and optimal solution value 169,532,339. For purposes of direct comparisons, the penalty algorithm RSD(P) of Staniec [Ref. 4] has been converted into our data structure and uses our computer program framework. RSD(P) has been improved by taking advantage of the conjugate gradient modifications in the master problem.

First, we will evaluate the performance of RSD(B) with different initial parameters in solving problem 4H. Subsequently, the comparisons between RSD(P) and RSD(B) are presented for problems 4H and 10H. Possible modifications of algorithm RSD(B) are then discussed. Finally, we test with a 100 commodity problem having more than one million variables and 300,000 constraints.

A. PERFORMANCE OF THE ALGORITHM RSD(B)

The following results were obtained on an IBM 3033 AP under VM/CMS. The Central Processor Unit (CPU) utilization time is used as a performance measurement. An "ϵ-optimality gap" is computed from the current upper and lower bounds as $(\overline{V} - \underline{V}) / cx^*$ or estimated as $(\overline{V} - \underline{V}) / \underline{V}$.

Since the barrier function decomposition requires an interior starting point, we will first investigate the response of RSD(B) to different starting points. Let c_{max} denote the maximum cost value in the network. While fixing $h^0 = 2 * c_{max}$, $a = 0.5$, and $r = 7$ in problem 4H, the choice of the parameter b establishing $\hat{y}' = b * \hat{y}$ resulted in the optimality gaps listed in Table 1:

	Gap (%)			
initial gap	66.76	51.93	39.42	26.95
iteration 7	10.21	9.87	9.56	10.49
iteration 16	4.60	3.98	4.60	5.06
iteration 24	3.35	2.25	3.05	2.73
iteration 32	2.03	1.42	1.53	1.86
iteration 40	1.19	0.99	1.28	1.38
iteration 48	0.75	0.68	0.89	0.89
reduction b	0.80	0.85	0.90	0.95

Table 1 : Response to Different Starting Points
Problem 4H

If the parameter b is selected too low, the dual estimates associated with the resulting interior starting point may not generate good, initial extreme points. If the starting value is chosen too high, the interior point gets too close to the boundary of the constraint region. The subsequent line search in the master problem is confined to a smaller search space of the constraint region resulting in slower convergence. Both values, $b = 0.85$ and $b = 0.9$, work well in the test problem. Further analysis will be based on $b = 0.9$. Good results with RSD were obtained for any value of r between 6 and 8.

The lower bounds obtained with the algorithm RSD(B) are sensitive to the initial parameter h^0. Recall that the dual estimates are computed as $\hat{\mu}_1 = h(Ax-b_1)^{-1}$. If the barrier parameter h^0 is relatively small, we approach the boundary very soon. As the slack on some jointly capacitated arcs almost vanishes, its reciprocal may take huge values resulting in extreme dual estimates. This situation leads to large oscillations in subsequent solutions of the subproblem. The phenomenum is demonstrated in Figure 3 where $h^0 = 0.5 * c_{max}$. As h^0 is increased, this effect seems to disappear, as shown in Figures 4 through 6. It is interesting to observe that some good lower bounds are obtained under all conditions. The extreme result obtained for the small value of h^0 suggests that $h^0 \geq c_{max}$ is the better choice.

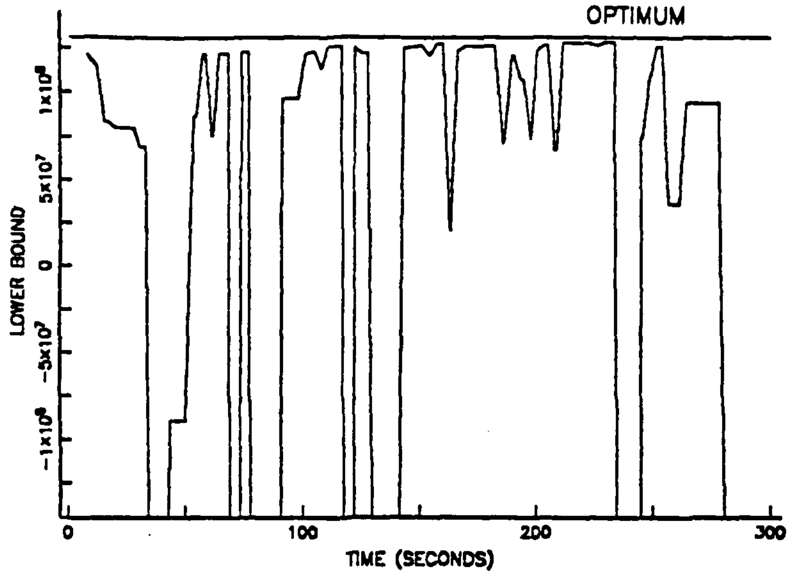


Figure 3: $\underline{V}(\hat{\mu}_1)$ for $h^0 = 0.5*c_{max}$, Problem 4H

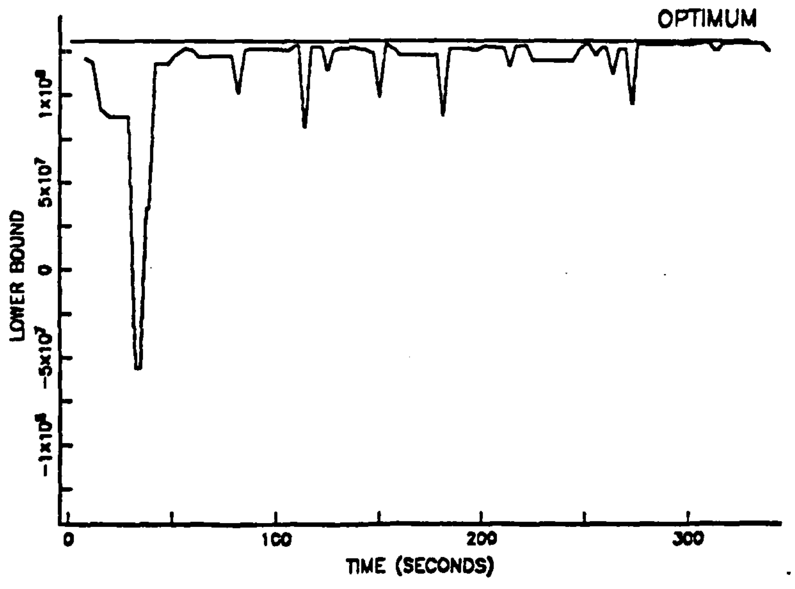


Figure 4: $\underline{V}(\hat{\mu}_1)$ for $h^0 = 1*c_{max}$, Problem 4H

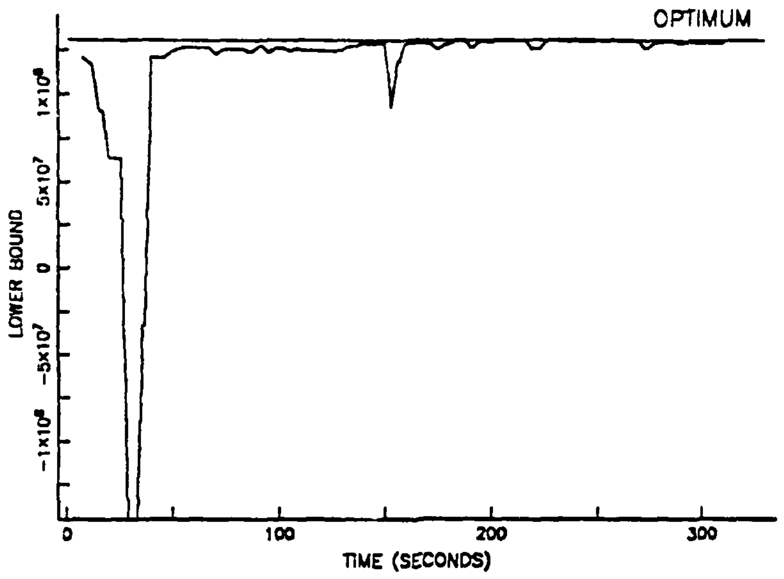


Figure 5: $\underline{V}(\hat{\mu}_1)$ for $h^0 = 1.5*c_{max}$, Problem 4H

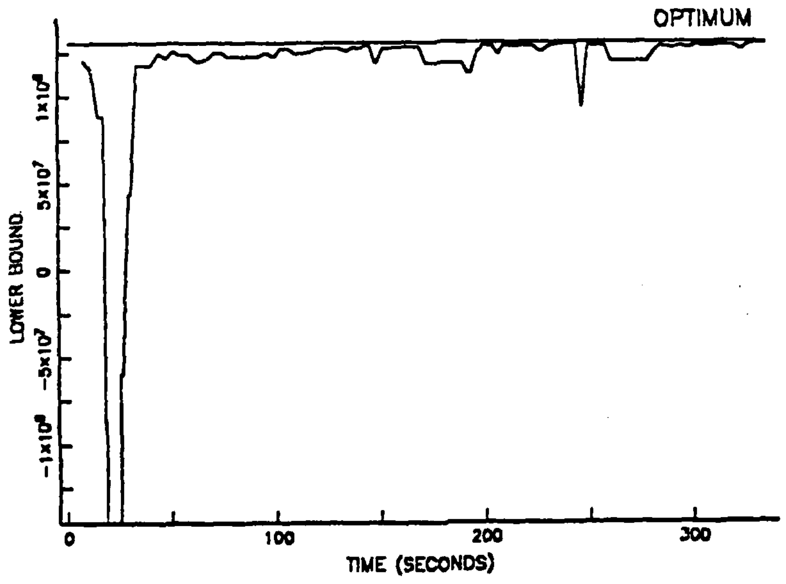


Figure 6: $\underline{V}(\hat{\mu}_1)$ for $h^0 = 2*c_{max}$, Problem 4H

If we want to select an initial value h^0 that balances the linear part cx and the barrier term in the objective function, we have to take into account that $|J_V|$ is not fixed but initially increasing. We do not have sufficient information about the size of the barrier term unless at least one pilot run has been conducted. Using the information that about 6% of the arcs are finally contained in the set J_V for problem 4H, we would obtain a value of $h^0 < 0.5 * c_{max}$, which is not the best choice, but may serve as a lower bound on h^0.

The differences obtained for the upper bounds are less significant. Figure 7 shows the sequence of values $c\hat{x}^k$ obtained for the same choices of h^0 as before.

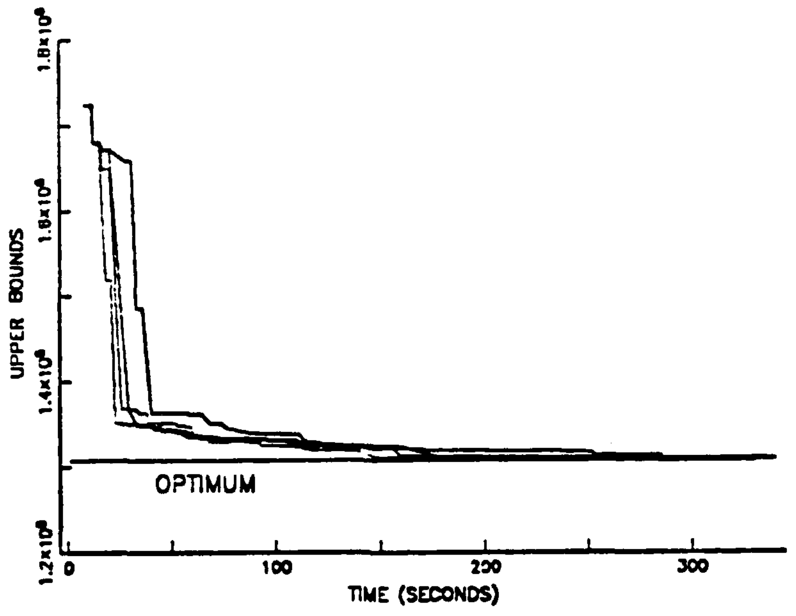


Figure 7: Response of $c\hat{x}^k$ to h^0, Problem 4H.

Any decrease in h creates a disturbance in the dual estimates. Since the master problem is not solved optimally for each value of h, the algorithm may not always provide better lower bounds before the next update of h occurs. We are still able to get good lower bounds when a relatively moderate rate of decrease a = 0.5 is used.

A solution trajectory for problem 4H is given in Figure 8 using $h^0 = 1.5 * c_{max}$, where a 0.6% optimal solution is obtained within 200 seconds. Instead of plotting the current upper and lower bounds, the current values of $\underline{V}(\hat{\mu})$ and $c\hat{x}^k$ are shown. We observe the almost strictly decreasing sequence of $c\hat{x}^k$ although the master problem is not solved optimally. The improved upper bounds obtained by the capacity allocation mechanism are denoted as cx' and indicate that significant gains are obtained only at the initial stage. The differences between $c\hat{x}^k$ and cx' nearly vanish after about 100 seconds although slightly tighter upper bounds are produced throughout.

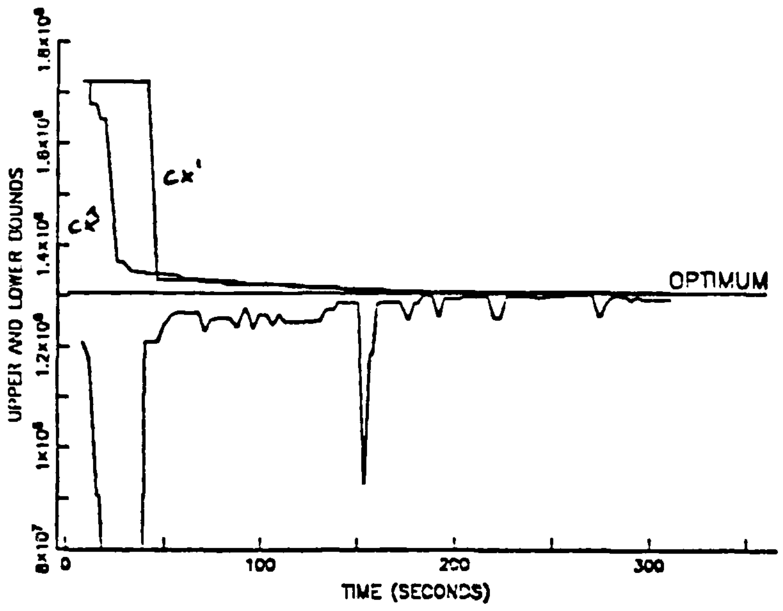


Figure 8: Trajectory of RSD(B), Problem 4H

It seems to be typical for the barrier function that good upper bounds are obtained at an early stage, whereas the lower bounds trail behind.

The trajectory of the objective value $B(h,\hat{x}^k)$ together with its linear part $c\hat{x}^k$ is given in Figure 9. We find that $B(h,\hat{x}^k)$ still approaches the optimum cx^* from below, the barrier term takes negative values over the whole range. The steps in its trajectory are due to the decreases of h by a factor of 0.5.

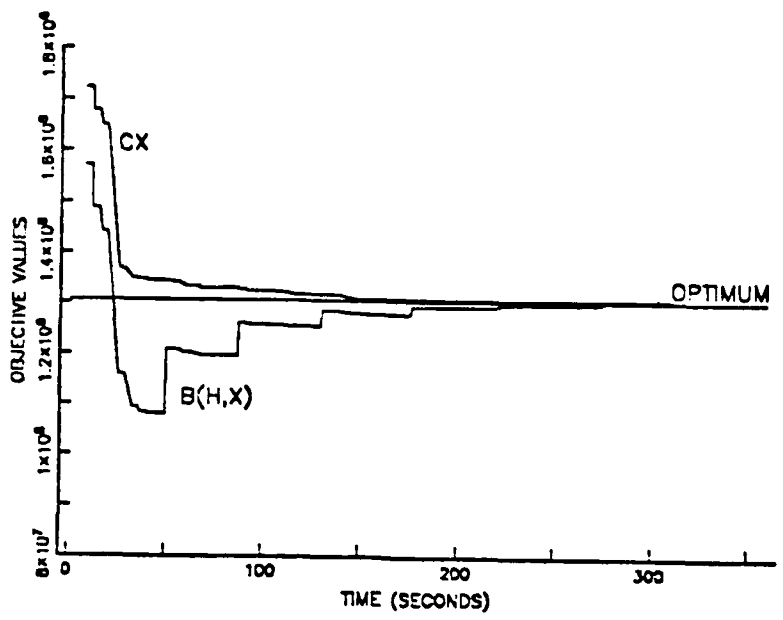


Figure 9: Values of the Objective Function, Problem 4H

A direct comparison between the algorithms RSD(P) and RSD(B) shows that the barrier function decomposition is less effective in the solution of the smaller problem 4H but is competitive in the problem 10H. The solutions to problem 4H are compared in Figure 10 were $h^0 = 0.001*c_{max}$ for RSD(P) and $h^0 = 1.5*c_{max}$ for RSD(B).

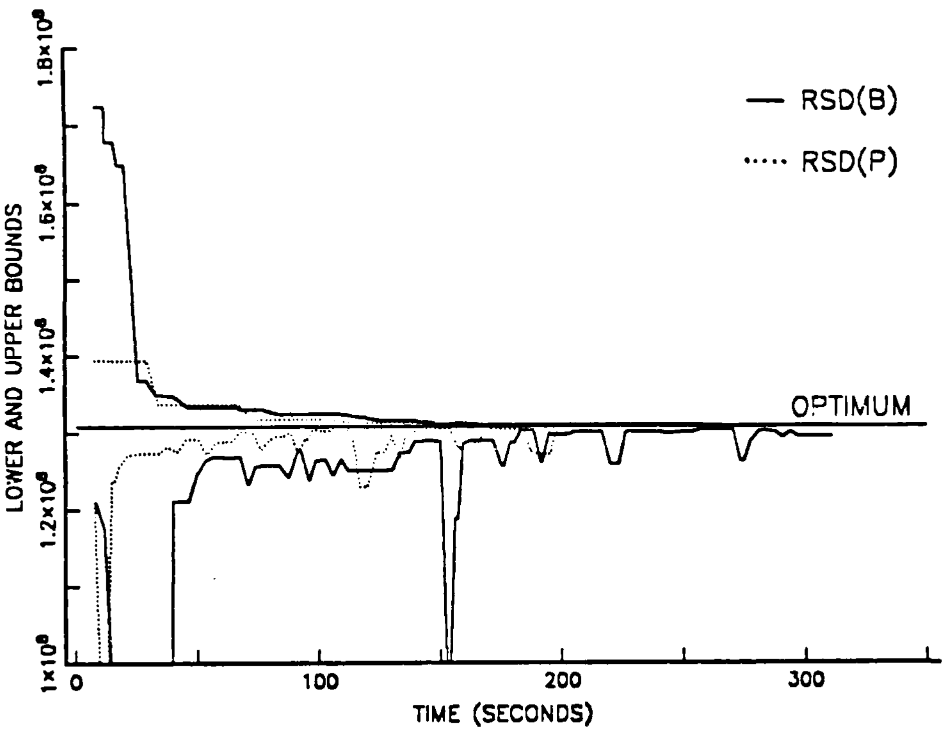


Figure 10: Comparison of RSD(P) to RSD(B), Problem 4H

If we use the same initial parameters for the solution of problem 10H, we obtain an interesting result. The trajectories are given in Figure 11. Obviously, the initial barrier and penalty parameters, respectively, are too low in both cases, yielding poor and oscillating lower bounds for RSD(B) versus bad upper bounds for RSD(P) due to insufficient penalty on the capacity violations. (This situation suggests investigation of a hybrid algorithm, incorporating both RSD(B) and RSD(P)).

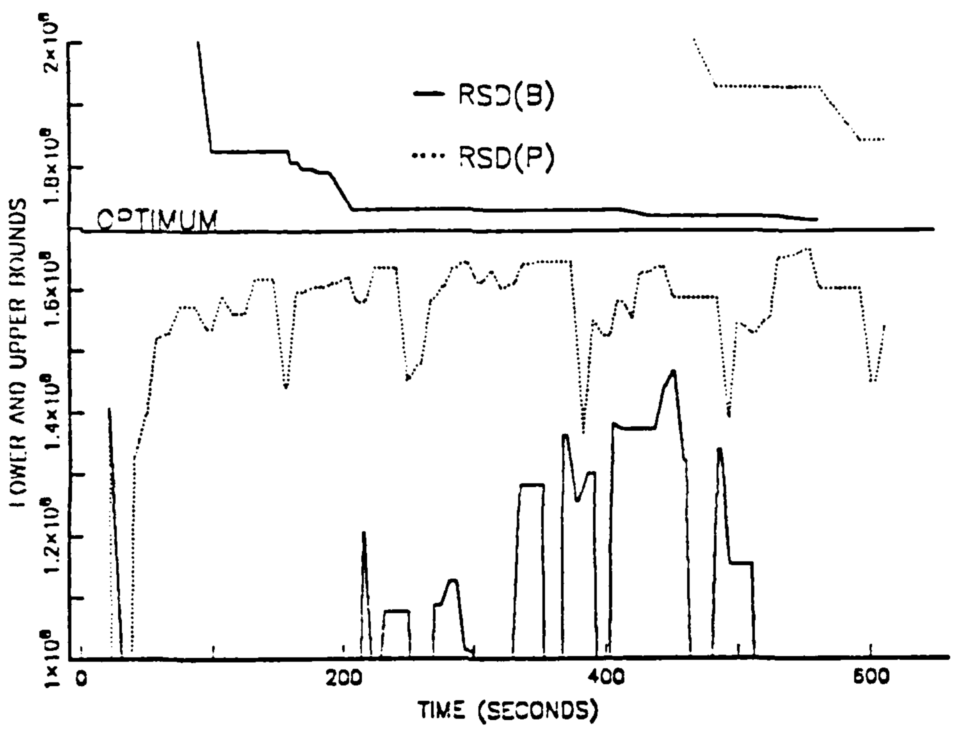


Figure 11: RSD(P) Versus RSD(B) , Problem 10H,
Same Initial Parameters

The initial parameter h^0 has been increased until good results are obtained for both methods. The result is shown in Figure 12. We observe that the final values of $\underline{V}(\hat{\mu}^k)$ decay for both algorithms. We do not generate improving lower bounds. Apparently, the parameter h is updated to rapidly before the master problem generates good dual estimates. Staniec [Ref. 4] reported similiar poor lower bounding for problem 10H.

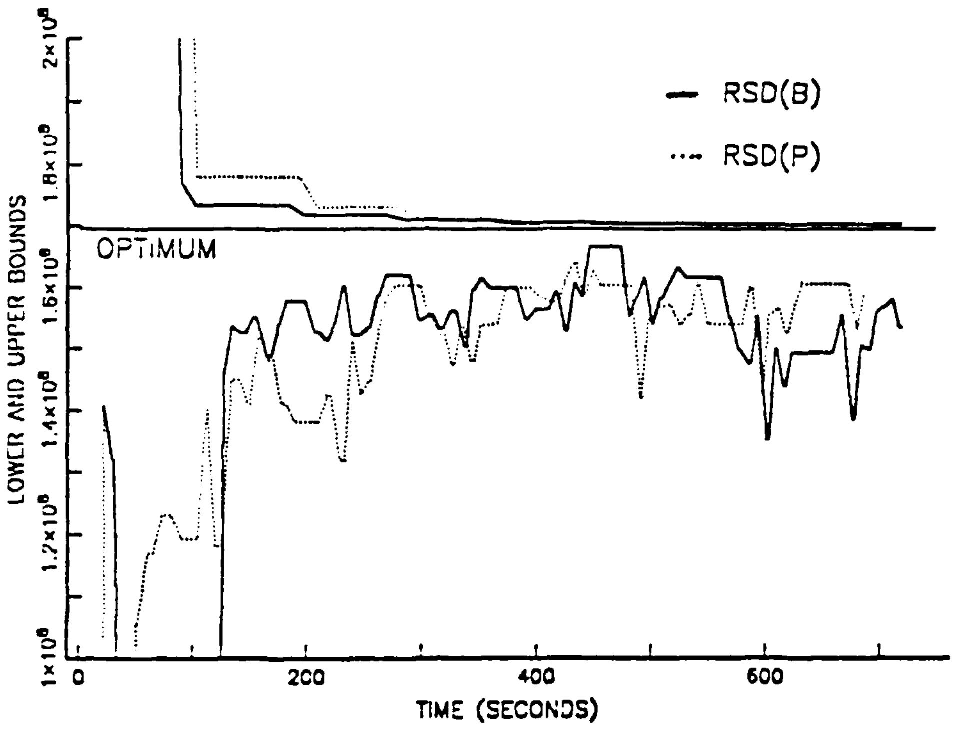


Figure 12: RSD(P) Versus RSD(B), Problem 10H, Best Parameters

In summary, we find that RSD(B) provides good upper bounds at each iteration and does not depend on the capacity allocation routine as the penalty algorithm does. On the other hand, the penalty algorithm provides better initial dual estimates, resulting in better lower bounds. Both algorithms converge to a good solution within a reasonable amount of time.

B. POSSIBLE MODIFICATIONS OF THE ALGORITHM

A first modification could be to postpone a further decrease of the barrier function parameter h until at least one better lower bound has been obtained compared to the bound generated for the previous value of h. Note, however, that the sequence of upper bounds is almost strictly decreasing from iteration to iteration and controlled by the magnitude of h. A higher barrier parameter results in higher upper bounds. Therefore, this choice should depend on the problem at hand. If a fast approximation of the solution is desired, h should be decreased more rapidly. If a higher resolution is required and sufficient computer resources are available, a better solution of the master problem is necessary, yielding better intermediate lower bounds.

Besides such a change of input parameters, a structural modification was also investigated. Since each capacity reallocation routine creates a subproblem solution that is feasible and at least as good as the current upper bound, this information can be passed to the master problem by using this solution as an additional "extreme" point. Experimentation on the test problem showed however that no significant improvements were obtained. This may be because the difference between the interior point solution of the master problem and the solution of the capacity reallocation procedure vanishes while the lower bounds are still trailing behind. Experimentation on other problems may yield different results.

The capacity allocation mechanism can be modified further. For RSD(B), capacity allocation for an interior point amounts to a redistribution of the available slack. As stated in equation (24), each commodity receives the same additional amount. A proportional allocation would be given by

$$\hat{y}_{pj} = \hat{x}_{pj} + \hat{x}_{pj} \, (b_1 - A\hat{x})_j \, / \, (A\hat{x})_j . \tag{71}$$

The disadvantage of this procedure is that a commodity with zero flow on that arc gets zero capacity allocated. To overcome this drawback, a convex combination of both methods is possible:

$$\hat{y}_{pj} = \hat{x}_{pj} + \beta_1 \, \hat{x}_{pj} \, (b_1 - A\hat{x})_j \, / \, (A\hat{x})_j + \beta_2 \, (b_1 - A\hat{x})_j \, / \, |P| \tag{72}$$

where $\beta_1, \beta_2 > 0$ and $\beta_1 + \beta_2 = 1$.

Experimentation with the test problem 4H showed improvements in the upper bounds in both cases, but since the lower bounds are still weak, the overall gain is not significant for this problem.

RSD(B) would be superior to RSD(P) if we could establish better lower bounds. The dual estimates $\hat{\mu}$ are a function of the barrier parameter h and the slack on the joint capacity arcs. Different dual estimates are obtained if we use a different value of h for each individual joint capacity arc. This approach leads to the weighted barrier function like the one proposed by Megiddo [Ref. 14] in general linear programming:

$$B(h,x,w) = cx - h \Sigma_j w_j \ln(b_1 - Ax)_j , \qquad (73)$$

where w is any real vector with positive components. Some experiments were done with different weights. All arcs that are violated in the initial solution with completely relaxed joint capacity constraints are more likely to be tight in the optimal solution. Therefore they are assigned a reduced weight to enable a faster approach to the boundary. Another weight factor used was proportional to the remaining slack on the corresponding arc. Unfortunately, neither attempt provided better solutions.

The final modification is a hybrid barrier-penalty algorithm. Starting with the barrier function decomposition in problem 10H, we shift to the penalty algorithm after 16 iterations using the extreme points generated so far. The main problem is the adjustment of the parameter h. Since we obtain good lower bounds for relatively small values of h, we reset h to the same initial value that we used in the independent approach. The results displayed in Figure 13 are unimpressive but further experimentation may improve results.

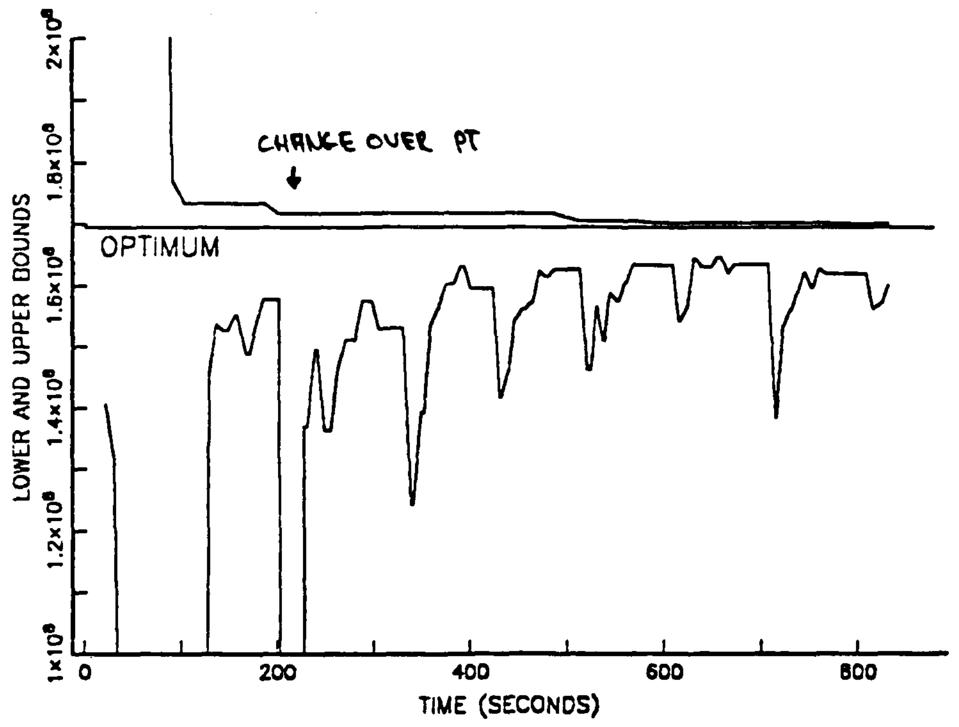


Figure 13 : Trajectory of the Hybrid Method

Finally, we obtain a solution to the 100 commodity network flow problem 100H which is about 3.5% optimal within 2650 seconds after 25 iterations and about 2.5% optimal within 3800 seconds after 38 iterations. The initial value h^0 has been increased to $h^0 = 35*c_{max}$. Staniec [Ref. 4] reports a solution to this problem obtained by a penalty method that achieved 4 % optimality after 1000 seconds and finished with 1.5% optimality after 3000 seconds. Our method did not show the same performance. The solution trajectory as shown in Figure 14 seems to indicate that the upper bounds are already very tight whereas the lower bounds are again poor. The objective function value (not shown) is still less than the lower bound, a further decrease of h is necesarry for convergence.

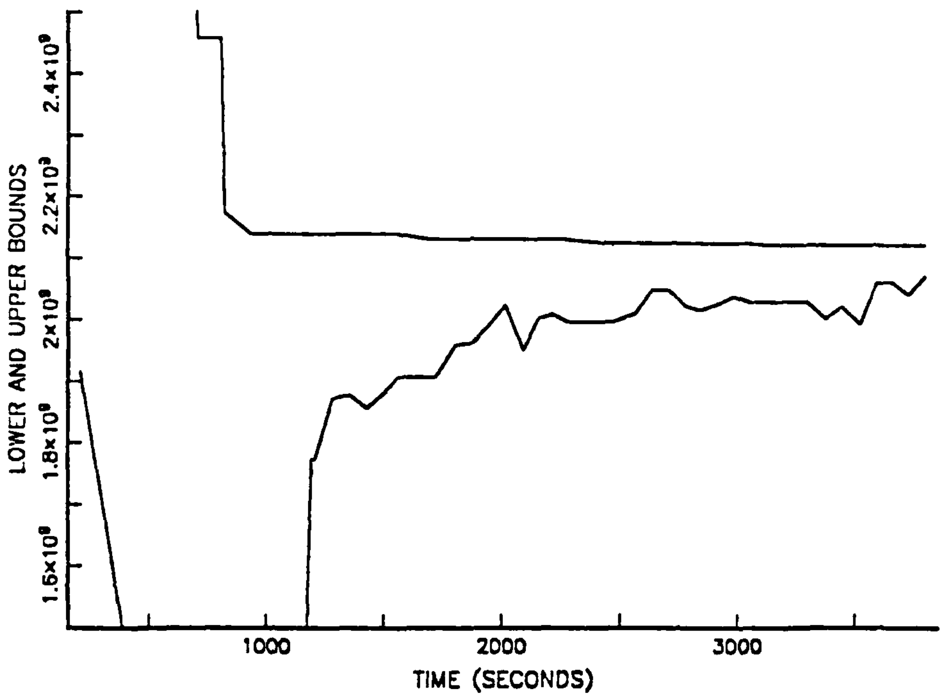


Figure 14 : Solution Trajectory, Problem 100H

The distribution of the CPU time between the master problem and the subproblems changes as we increase the number of commodities. Although the solution of each pure network flow problem requires less than one second CPU time per commodity in the average for all test problems, we find that the subproblems are expensive to solve and consume the largest portion of the total CPU time. This distribution is shown in Table 2. The amount of time not used by the master problem or the subproblem is mostly consumed in the input/output operations.

	Problem 4H	Problem 10H	Problem 100H
Master Problem	25.5%	11.2%	7.1%
Subproblem	49.9%	66.9%	86.3%
Other	24.6%	21.9%	6.6%

Table 2: Distribution of CPU Time for Test Problems of Increasing Size

However, the subproblem solves for each commodity independently and more than one pure network flow problem could be solved at a time. This feature makes our method highly parallel with a potential reduction of nearly $1/|P|$ in elapsed computation time for the subproblem.

V. CONCLUSIONS

The method of logarithmic barrier function decomposition is a useful tool for the solution of large-scale multicommodity flow problems. The algorithm RSD(B) is a variant of the price-directive decomposition method. It generates a sequence of interior points which provide intermediate feasible solutions while converging towards the optimum. The technique of restricted simplicial decomposition (RSD) proves to be useful in the solution of the nonlinear master problem. However, RSD(B) seems to be robust and does not require an optimal solution to the master problem. It is competitive with penalty decomposition methods and relatively easy to implement. Since it achieves good feasible solutions at an early stage, it may be used as a starting technique in other algorithms like the hybrid barrier-penalty technique. The use of RSD(B) in other large-scale multicommodity flow problems is recommended.

LIST OF REFERENCES

1. Bradley, G. H.,Brown, G. G., and Graves, G. W., "Design and Implementation of Large Scale Primal Transshipment Algorithms", Management Science, Vol. 24-1, 1977, pp. 1-34.

2. Kennington, J., "A Survey of Linear Cost Multi-commodity Network Flows", Operations Research, Vol. 26, 1978, pp. 209-236.

3. Assad, A.,"Multicommodity Network Flows--A Survey", Networks, Vol. 8, 1978, pp. 37-91.

4. Staniec, C. J., Solving the Multicommodity Transshipment Problem, Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 1987.

5. Kennington, J. L., Helgason, R. V., Algorithms for Network Programming, John Wiley & Sons, New York, 1980.

6. Fisher, M. L., "An Application's Oriented Guide to Lagrangian Relaxation", Interfaces, Vol. 15-2, 1985, pp. 10-21.

7. Goffin, J. L., "On Convergence Rates of Subgradient Optimization Methods", Mathematical Programming, Vol.13, 1977, pp. 329-347.

8. Sandi, C., "Subgradient Optimization", Chapter 3 in Combinatorial Optimization, ed. Christofides, N., John Wiley & Sons, New York, 1979.

9. Kelly, J. E., "The Cutting Plane Method for Solving Convex Programs", SIAM J. Appl. Math, Vol. 8, 1960 pp. 703-712.

10. Luenberger, D. G., Linear and Nonlinear Programming, 2nd ed., Addison-Wesley, Menlo Park, California, 1984.

11. Fiacco, A. V., and McCormick, G. P., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley & Sons, New York, 1968.

12. Bazaraa, M.S., and Shetty, C. M., Nonlinear Programming, Theory and Algorithms, John Wiley & Sons, New York, 1979.

13. Frisch, K. R., "The Logarithmic Potential Method of Convex Programming", unpublished manuscript, University Institute of Economics, Oslo, 1955.

14. Megiddo, N., Pathways to the Optimal Set in Linear Programming, Preliminary Report, IBM Almaden Research Center, San Jose,California, and Tel Aviv University, Israel 1986.

15. Gill, P. E., Murray, W., Saunders, M. A., Tomlin, J. A., and Wright, M. H., On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method, Systems Optimization Labratory Technical Report SOL 85-11, 1985.

16. Karmarkar, N. K., "A New Polynomial-Time Algorithm for Linear Programming", Combinatorica, Vol.2, 1984, pp. 373-395.

17. Hearn, D. W., Lawphongpanich, S., and Ventura,J. A., "Restricted Simplicial Decomposition:Computation and Extensions, Mathematical Programming Study, Vol. 31, 1987, pp. 99-118.

18. Staniec, C. J., Design and Solution of an Ammunition Distribution Model by a Resource-Directive Multi-commodity Network Flow Algorithm, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1984.

19. Wright, M. A., Numerical Methods for Nonlinear Constrained Optimization, Ph.D. Dissertation, Stanford University, Stanford, California, March 1976.

20. Mifflin, R., "On the Convergence of the Logarithmic Barrier Function", in Numerical Methods for Non-linear Optimization, ed. Lootsma, F. A., Academic Press, London, 1972.

21. Ryan, D. M., "Penalty and Barrier Functions", Chapter X in Gill, P. E., and Murray, W. , Numerical Methods for Constrained Optimization, Academic Press, London, 1974.

22. Lootsma, F. A., "A Survey of Methods for Solving Constrained Minimization Problems via Unconstrained Minimization", in Numerical Methods for Nonlinear Optimization, ed. Lootsma, F. A., Academic Press, London, 1972.

23. Frank, M. and Wolfe, P., "An Algorithm for Quadratic Programming", Naval Research Logistics Quarterly, Vol.3, 1956, pp. 95-110.

24. Wolfe, P., "Convergence Theory in Nonlinear Programming", in Integer and Nonlinear Programming, ed. Abadie, J., American Elsevier, New York, 1970.

25. Geoffrion, A. M., "Elements of Large-Scale Mathematical Programming", Management Science, Vol.16, 1970, pp. 652-691.

26. Von Hohenbalken, B., "A Finite Algorithm to Maximize Certain Pseudoconcave Functions on Polytopes", Mathematical Programming, Vol. 8, 1975, pp. 189-206.

27. Holloway, C. A., "An Extension of the Frank and Wolfe Method of Feasible Directions", Mathematical Programming, Vol.6, 1974, pp.14-27.

28. Wolfe, P., "Methods for Nonlinear Programming", in Recent Advances in Mathematical Programming, ed. Graves, R. L., and Wolfe, P., 1963.

29. McCormick, G. P., "Anti-Zig-Zagging by Bending", Management Science, Vol.15, 1969, pp. 315-320.

30. Reklaitis, G. V., Ravindran, A., and Ragsdell, K.M., Engineering Optimization, John Wiley & Sons, New York, 1983.

31. MacLennan, B. J., Principles of Programming Languages: Design, Evaluation, and Implementation, 2nd ed., Holt,Rinehart and Winston, New York 1987.

32. Brown, G. G., Graves, G. W., XS Mathematical Programming System, perpetual working paper, 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Professor R. Kevin Wood, Code 55Wd Department of Operations Research Naval Postgraduate School Monterey, California 93943	15
4.	Professor Gerald G. Brown, Code 55Bw Department of Operations Research Naval Postgraduate School Monterey, California 93943	1
5.	Lieutenant Commander Heinrich Lange German Fast Patrol Boat Flotilla Muerwiker Strasse D-2390 Flensburg, West Germany	10